1장. 코틀린 이해

1.1. 코틀린이란

1.1.1. 코틀린 소개

- 스칼라(Scala, 2003), 고(Go, 2009), 다트(Dart, 2011), 스위프트(Swift ,2014)
- 젯브레인(https://www.jetbrains.com/idea/)의 오픈소스 그룹에서 개발
- 2011년이었으나 정식 버전은 2016년에 발표
- 2017년 5월 구글 I/O 행사에서 안드로이드의 공식 언어로 코틀린을 지정

1.1.2. 코틀린 특징

- 자바, 안드로이드 100% 호환
- 자바, 안드로이드, 브라우저, 네이티브 애플리케이션 개발
- Intellij, 안드로이드 스튜디오, 이클립스, CLI 등을 통한 개발
- 함수형 언어, Lambdas, Extension, Null Safety 등 최신 언어의 트랜드 지원

1.2. 코틀린으로 작성 가능한 프로그램

1.2.1. 자바를 코틀린으로

```
class User {
    private int no;
    private String name;
    private String  email;
    public int getNo() {
        return no;
    }
    public void setNo(int no) {
        this.no = no;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}
```

```
data class User(val no: Int, val name: String, val email: String)
```

1.2. 코틀린으로 작성 가능한 프로그램

1.2.2. 안드로이드를 코틀린으로

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button testBtn=(Button)findViewById(R.id.testBtn);
        testBtn.setOnClickListener(this);
    }
    @Override
    public void onClick(View view) {
        Toast.makeText(this, "button click!!", Toast.LENGTH_SHORT).show();
    }
}
```

```
class MainActivity : AppCompatActivity(), View.OnClickListener{
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        testBtn.setOnClickListener(this)
    }
    override fun onClick(p0: View?) {
        Toast.makeText(this, "Button Click!!", Toast.LENGTH_SHORT).show()
    }
}
```

1.2. 코틀린으로 작성 가능한 프로그램

1.2.3. 자바스크립트를 코틀린으로

```
window.onload=function(){
  var btn=document.getElementById("btn");
  var result=document.getElementById("result");
  btn.addEventListener('click', function(){
    result.innerText='Hello Kkang';
  });
}
```

```
fun onLoad(){
    val btn = document.getElementById("btn")!! as HTMLElement
    val result=document.getElementById("result")!! as HTMLElement
    btn.onclick={
        result.appendText("Hello Kkang")
    }
}

fun main(args: Array<String>) {
    window.onload = {
        onLoad()
    }
}
```

1.2. 코틀린으로 작성 가능한 프로그램

1.2.4. 서버 측 웹 애플리케이션을 코틀린으로

```
@WebServlet(name = "Hello", value = "/hello")
public class HelloServlet extends HttpServlet {
    public void doGet(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse)
            throws IOException {

        String name = httpServletRequest.getParameter("name");

        httpServletResponse.getWriter().print("Hello "+name);
    }
}
```

```
@WebServlet(name = "HelloKotlin", value = "/helloKotlin")
class TestKotlinServlet: HttpServlet() {
    override fun doGet(req: HttpServletRequest, res: HttpServletResponse) {
        val name=req.getParameter("name")
        res.writer.write("Hello Kotlin "+name)
    }
}
```

1.2. 코틀린으로 작성 가능한 프로그램

1.2.5. 스프링 부트를 이용하는 Restful 서비스를 코틀린으로

```
@RestController
public class GreetingController {

    AtomicLong counter=new AtomicLong();

    @GetMapping("/greeting")
    public Greeting greeting(@RequestParam(value="name", defaultValue = "World") String name){
        return new Greeting(counter.incrementAndGet(), "Java, "+name);
    }
}
```

```
@RestController
class GreetingController {

    val counter = AtomicLong()

    @GetMapping("/greeting")
    fun greeting(@RequestParam(value = "name", defaultValue = "World") name: String) =
            Greeting(counter.incrementAndGet(), "Hello, $name")

}
```